OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12)(19)(CA) **Demande-Application**

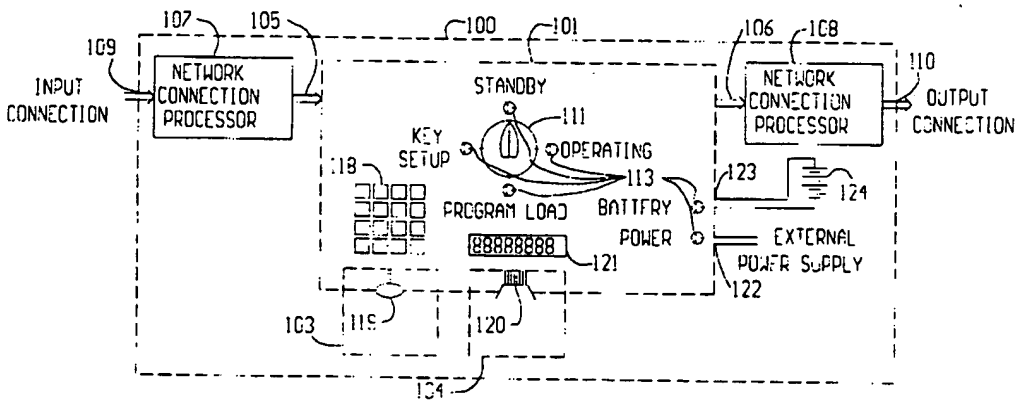
(21)(A1) **2,271,178**
(22) 1999/05/06
(43) 1999/07/06

(72) MOREAU, Thierry, CA
(71) CONNOTECH EXPERTS-CONSEILS INC., CA

(51) Int.Cl.[6] H04L 9/30

(54) **APPAREIL CRYPTOGRAPHIQUE A CLE REVELEE COTE-SERVEUR AVEC CLES SECRETES DE PROTECTION ET D'ISOLEMENT DE RESEAUX PUBLICS**

(54) **SERVER-SIDE PUBLIC KEY CRYPTOGRAPHY APPARATUS WITH PRIVATE KEY PROTECTION AND ISOLATION FROM PUBLIC NETWORKS**



(57) A server-server public-key cryptography apparatus is disclosed for use in the computer facilities of central organizations, e.g. on-line service providers. The apparatus has two network connections of the type common to computer networks, used respectively for exclusively receiving input data and exclusively transmitting output data for some elementary private key computation (digital signature, public key decryption, secret key establishment primitive based on a public key algorithm). The secrecy of this private key is supported by a number of the present invention features. Among others, are provided the cryptographic key management operations needed to initially configure, operate, maintain, and re-install in the case of disaster recovery. In operations, the access to the elementary private key computation has to be restricted to those computer applications that are the legitimate users of the private key. The one-way input connection, the one-way output connection, and some features of the cryptographic key management operations are provided to secure this restricted access to the function performed by the secure computing device.

ABSTRACT

A server-server public-key cryptography apparatus is disclosed for use in the computer facilities of central organizations, e.g. on-line service providers. The apparatus has two network connections of the type common to computer networks, used respectively for exclusively receiving input data and exclusively transmitting output data for some elementary private key computation (digital signature, public key decryption, secret key establishment primitive based on a public key algorithm). The secrecy of this private key is supported by a number of the present invention features. Among others, are provided the cryptographic key management operations needed to initially configure, operate, maintain, and re-install in the case of disaster recovery. In operations, the access to the elementary private key computation has to be restricted to those computer applications that are the legitimate users of the private key. The one-way input connection, the one-way output connection, and some features of the cryptographic key management operations are provided to secure this restricted access to the function performed by the secure computing device.

Server-side Public Key Cryptography Apparatus with Private Key Protection and Isolation from Public Networks

Field of the Invention

The present invention relates to a digital computing apparatus used by a central organization to perform the mathematical operation possible only with the knowledge of the private key in the context of an asymmetric public/private key pair cryptosystem.

Description of the Prior Art

The implementation of public key cryptography often requires a central organization to generate a public/private key pair, distribute the public key, and preserve the secrecy of the private key with an very high level of security. During the normal operation of the system, the central organization, as a critical part of some secure application protocol, routinely performs a critical mathematical operation possible (that is, computationally within the reach of modern computers) only with the knowledge of the private key.

Generally, there are three distinct application patterns where public key cryptography apparatuses may be needed:

(a) The one considered by the present invention, in which a server has to perform the computation made possible only by the presence of some private key inside the apparatus, at least for the duration of the computation. Typically, such servers are installed in a data processing center of a medium or large organization, or within the facilities of an on-line service

supplier. Confidence in the secrecy of the private key can be assisted, to some extent, by tasks assigned to employees responsible for overseeing computer operations.

(b) For the routine verification of digital signatures and security certificates, throughput is very important but there is no private key that deserves confidentiality protection (digital signature verification requires the public key only).

(c) For non-repudiation service at the end-user level (e.g. a digital signature smartcard), interesting design characteristics are very small size and low power requirements combined with tamper-resistance, all of this at a very low unit cost. This application pattern requires a hiding place for a private key, but no accountable employee is around to assist security, e.g. participate in control functions (as in the application pattern considered by the present invention).

An application area for the present invention is a Certification Authority (CA) in the context of a Public Key Infrastructure (PKI), in which case the critical mathematical operation is a digital signature generation. The PKI environment is well documented in the prior art. Although the case of a CA is famous, there are other cases where a server routinely performs a public key digital signature. Such a case is disclosed in US patent document 5,604,801, in which a signature authorization protocol controls the digital signature operation done within encapsulation of a signature server. See also the article by Christopher J. Holloway, "Controlling Digital Signatures Services Using A Smartcard", in Computers & Security, Vol. 14(1995) pp.681-690.

US patent document 5,604,801, Dolan, George M., Holloway, Christopher J., Matyas, Jr., Stephen M., *Public key data communications system under control of a portable security device*, Feb. 18, 1997, (application number 383129, Feb. 3, 1995), assigned to International Business Machines Corporation

Holloway, Christopher J., *Controlling Digital Signatures Services Using A Smartcard*, Computers & Security, Vol. 14(1995) pp.681-690

Other application areas for the present invention are various schemes where a central organization decrypts ciphertext messages received from public networks, in which case the mathematical operation is a public key decryption. One such example is the public key decryption done by the server in the Beller-Yacobi protocols, specifically in the "real-time protocol execution" phase in U.S. patents 5,299,263 and 5,406,628.

US patent document 5,299,263, Beller, Michael J., Yacobi, Yacov, *Two-Way Public Key Authentication and Key Agreement for Low-cost Terminals*, March 29, 1994 (application number 26,673, March 4, 1993), assigned to Bell Communications Research, Inc.

US patent document 5,406,628, Beller, Michael J., Yacobi, Yacov, *Public Key Authentication and Key Agreement for Low-cost Terminals*, April 11, 1995 (application number 101,437, August 2, 1993), assigned to Bell Communications Research, Inc.

Yet other application areas for the present invention are secret key establishment protocols that are based on public key cryptography. Secret key establishment can be seen as a superset of public key decryption. See for example US patent document 4,771,461, US patent document 5,142,578, US patent document 5,784,463, and PCT International application number 9852316A1, by the present inventor. In the PCT International application number 9852316A1, three different public key cryptography schemes are presented, and each one embeds some critical mathematical operation that can conveniently be performed by the present invention.

US patent document 4,771,461, Matyas, Stephen M., *Initialization of Cryptographic Variables in an EFT/POS Network with a Large Number of Terminals*, September 13, 1988, (application number 879,784, June 27, 1986)

US patent document 5,142,578, Matyas, Stephen M., Johnson, Donald B., Le, An

V., Prymak, Rostislaw, Wilkins, John D., Martin, William C., Rohland, William S., *Hybrid public key algorithm/data encryption algorithm key distribution method based on control vectors*, Aug. 25, 1992, (application number 748407, Aug. 22, 1991), assigned to International Business Machines Corporation

US patent document 5,784,463, Chen, James F., Wang, Jieh-Shan, *Token Distribution, Registration, and Dynamic Configuration of User Entitlement for an Application Level Security System and Method*, July 21, 1998 (application number 760,414, December 4, 1996), assigned to V-ONE Corporation

Moreau, Thierry, *Initial Secret Key Establishment Including Facilities for Verification of Identity*, Patent Cooperation Treaty (PCT) International application number 9852316A1 (PCT/CA 98/ 00431), filed on May 7th, 1998, priority date based on provisional U.S. patent application number 60/046.047, filed on May 9th, 1997, published by the PCT International Bureau on November 19th, 1998, CONNOTECH Experts-conseils Inc., Montréal, Canada

The published standard FIPS 140-1 specifies some requirements for physical and logical protection of cryptographic apparatuses, while the Working Draft ANSI X9.30-199x Part 3ANSI/ABA specifies some requirements for the key management aspect of digital signatures for Certification Authorities. These publications are not meant to teach how to build a cryptographic apparatus, but rather what level of security should be provided. This is not unlike the US patent 5,208,858 (Siemens) which refers (in general terms) to "structural space (room)" protection that encapsulates a "central server" and a "code algorithm computer."

U.S. Department of Commerce, National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-1, reference FIPS PUB 140-1, 1994 January 11

Accredited Standards Committee X9-Financial Services (ANSI ASC X9) Working Draft ANSI X9.30-199x *Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry: Part 3: Certificate Management for DSA*, American Bankers Association, Washington, DC, November 19, 1994 (document N24-94)

US patent document 5,208,858, Vollert, Emmerman, Wildgrube, Eberhard, *Method for Allocating Useful Data to a Specific Originator*, May 4, 1993 (application number 645,496, January 24, 1991), assigned to Siemens Aktiengesellschaft, Munich, Fed. Rep. of Germany

A secure computer architecture is disclosed in US patent 5,237,616. See also the article *Transaction System Security*, IBM Systems Journal, Vol. 30, No. 2, 1991, pp 206-229. In this architecture, cleartext secret key material travels between at least three integrated circuits within a secure boundary. It is considered more secure to confine the cleartext cryptographic key material to a single integrated circuit that encompasses internal memory and sufficient processing power for the complete cryptographic processing of a complete transaction. The US patent document 5,551,051 is an example of a very involved processor context isolation scheme, which appear more suited to high reliability systems than to cryptographic computations using a secret key.

US patent document 5,237,616, Abraham, Dennis G., Aden, Steven G., *Secure computer system having privileged and unprivileged memories*, Aug. 17, 1993, (application number 947871, Sep. 21, 1992), assigned to International Business Machines Corporation

Abraham, Dennis G., Dolan, George M., Double, G. P., Stevens, J. V., Transaction System Security, IBM Systems Journal, Vol. 30, No. 2, 1991, pp 206-229

US patent document 5,551,051, Silverthorn, Lee, Cornils, Curtis, Kirchner, Mark L.,

Stephens, Susan D., Crouse, Parker E., *Isolated multiprocessing system having tracking circuit for verifying only that the processor is executing set of entry instructions upon initiation of the system controller program*, Aug. 27, 1996, (application number 309379, Sep. 20, 1994), assigned to Motorola, Inc.

One of the difficulty with current designs of secure cryptographic apparatuses is their single point of connection with a host system. Then, the host system might be programmed to exploit any weakness in the cryptographic apparatus, and this creates a requirement for host system security measures. This is becoming more and more critical as the on-line services are exposed to the hacking pressure from the Internet. Also, with public key cryptography, if the algorithm implemented in the cryptographic apparatus is (or might be) vulnerable to the so-called "chosen ciphertext attack," then the single point of connection with a host system becomes a door open to cryptanalysis attempts.

The prior art misses a comprehensive disclosure for a cryptographic apparatus in the application areas addressed by the present invention. To be useful to someone skilled in the art of cryptographic system implementation, the disclosure of a public key cryptography apparatus for the purpose mentioned above should include, among other things, details about a) key management principles, b) security measures for the interface between the apparatus and the other computing environments that participate in the delivery of the overall application, and c) physical characteristics that are related to security of key management and protocols.

Key management principles are disclosed in the US patent 5,214,698, but without reference to the security of public key cryptography operations. The US patent 5,214,698 deals with the secure input of cryptographic keys in an apparatus.

US patent document 5,214,698, Smith, Ronald M., Yeh, Phil C., Easter, Randell J., Johnson, Donald B., Le, An Van, Matyas, Stephen M., Thomas, Julian, Wilkins, John D., *Method and Apparatus for Validating Entry of Cryptographic Keys*, May 25,

1993 (application number 672,265, March 20, 1991), assigned to International Business Machines

The USA patent 5,675,649 focuses on key generation and safekeeping. This last US patent 5,675,649 is intended for public key cryptography schemes (e.g. RSA) where key generation is done by complex mathematical algorithms implemented in software on a general purpose computer. The US patent 5,675,649 goes further and uses the mathematical trick (the so-called Shamir secret sharing scheme) for private key safekeeping. Involved public key generation algorithms are discussed at length in the article by Ueli M. Maurer, "Fast Generation of Prime Numbers and Secure Public-Key Cryptography Parameters," in Journal of Cryptology (1995) Vol 8, pp 123-155.

US patent document 5,675,649, Brennan, J.J., Geist, Bruce K., Van Eeuwen, Jeffrey A., *Process for Cryptographic Key Generation and Safekeeping*, October 7, 1997, (application number 565,525, November 30, 1995), assigned to Electronic Data Systems Corporation

Maurer, Ueli M., *Fast Generation of Prime Numbers and Secure Public-Key Cryptography Parameters*, Journal of Cryptology (1995) Vol 8, pp 123-155

To many observers of the field of information security, the actual market demand for information security solutions has always been much less that the growth forecasts that could be inferred from the evolving conditions. This can be imparted mainly to organizational inertia when faced with a business function, information security, that has both an insurance-like benefit pattern and a high technology contents. This fact of life has to be taken into account to appraise the inadequacies of the prior art. This general statement can be refined and augmented for the field of the present invention. So we list hereafter some real-world constraints that should be obvious to those who has been exposed to actual cryptographic key management in medium and large organizations for which the information security function

should be taken seriously.

- Costs must be reasonable. Off-the-shelf components are preferred to custom, security-enhanced components. Unit production cost and amortization of development cost must be within cost-effectiveness range.

- Behind closed doors, expect carelessness in security systems operations. Large organizations like financial institutions are formed by complex interrelations of motivations and personalities, and their public image is seldom aligned with the inside culture. The security procedures taught in textbook often don't fit an organizational motivation for expeditiousness. Usually, the public relations department is relied upon to manage any embarrassment due to the disclosure of lax internal controls.

- Education is scarce. With involved security schemes, very few individuals feel genuinely confident about the underlying theory of operation (besides those directly involved in the original design). When the behavioral integrity of employees and agents is critical, so becomes education and training.

- Resources devoted to security are short-term, much shorter than the life-cycle of a security application. This is a consequence of a general trend in business environment. An investment decision brings resources for a the short implementation phase of a project, while the operating budgets are recurrently reviewed for apparent inefficiencies. Don't expect qualified personnel and trusted agents to remain in place for long periods. When a security breach occurs, it may be very difficult to get hold of critical personnel, relevant documentation, or computer program source code.

- Expect major upgrades or conversions within the lifetime of a security application. Large service organizations often operate with a number of automated systems that were acquired throughout decades and that are interconnected, physically or logically, in a

closely knit network. Changing one system often has unexpected impact on many other ones, such as forced major upgrades or conversions.

- Some compromise or concession will be forced on security scheme implementations. Competition among service organizations force the early fielding of systems, making security shortcuts unavoidable. Also, mere operational constraints can have the same effect.

As an example, in view of these principles, the process disclosed in the before mentioned US patent 5,675,649 appears of limited applicability. It is hard to find skilled agents able to review the software source code used in a public key generation process. It is doubtful that the computer shutdown procedure be implemented with all due diligence. The disk media on which the key shares are to be kept must be reliable for extended periods of storage. Some organizations are deemed to mismanage the Shamir secret sharing scheme, e.g. in a two out of four scheme, by putting two shares in a main safe storage and the other two shares in a backup safe storage (if not simply all four shares in the two storage locations). For all practical purposes, sharing a key in more than two shares may be ineffective: with a larger number of trusted persons, someone will be responsible for coordination, and this role becomes a single point of failure in the so-called "social engineering" attacks.

It is thus important for a key management scheme to be effective in spite of typical real-world constraints on daily operations. Critical secret keys must be properly protected by a secure computing device and a related key management scheme. The private key of a central organization is indeed very critical. The idea is to be realistic about the human factor being the weakest link of the chain, while keeping in mind the more intricate vulnerabilities of public key cryptography.

The first lines of defense for the secrecy protection of the private key are the installation of the apparatus in a controlled computer room, and the storage of private key components in safe

boxes.

Summary of the Invention

The two representative uses of a central organization private/public key pair are well known. They are, respectively, 1) the digital signature operation, and 2) the public key decryption of short cryptograms, usually to establish a secret key in a hybrid cryptosystem. Indeed, the present invention addresses equally these two application areas, namely digital signatures and secret key establishment, and the details of the elementary private key computation in each application area are well known to the field. The present invention more specifically addresses the demanding security requirements for a server system private key when the corresponding public key is widely distributed, used, and relied upon. In these circumstances, the long-term protection of the private key secrecy is critical because if the private key would be compromised then the forced replacement of the widely distributed public key is a significant nuisance.

Either use of a central organization's private key (digital signature or secret key establishment) can be viewed as a step in an administrative process. In this perspective, the present invention is based on the realization that the elementary private key computation is a processing step that either drastically lowers the vulnerability to data tampering (it's the digital signature that definitely sets the seal on e.g. the details of a security certificate) or drastically raises the level of data sensitivity (e.g. a secret key just recovered by the elementary private key computation is definitely sensitive). Accordingly, there is one side of the elementary private key computation where individual transaction security is very important, and the opposite side where controls are much less important.

To be more specific, for the digital signature application, there are a number of verifications and controls to be done upstream of each elementary private key computation (digital signature operation), hence a requirement for secure access control protection for the "final approval of

digital signature release" targeted at a specific set of transaction data (e.g. the final approval of a security certificate by a CA in the PKI context). Secure access control protection may be achieved with a secret-key MAC (Message authentication code) that is verified by the secure cryptographic device (SCD) before actually performing the digital signature operation, but other cryptographic protection for authentication, and/or accountability may be employed (e.g. the signature authorization protocol disclosed in the before mentioned US patent document 5,604,801).

Conversely, for the secret key establishment application, some data processing application downstream of the elementary private key computation must process each recovered secret key with adequate confidentiality protection. For instance, in the PCT International application number 9852316A1, the out-of-band verification of registrant's identity requires confidentiality protection of registration data from the point of public key decryption to the actual verification of identity. In the preferred embodiment of the present invention, conventional application-level confidentiality protection is applied with secret-key encryption from the point of public key decryption, namely the secure cryptographic device (SCD), to the application computing environment.

The present invention isolates the private key in a secure cryptographic device (SCD) with two separate data communications connections, respectively a one-way input connection and a one-way output connection. The SCD has built-in physical security protection, notably the absence of any external electrical contacts where a logic analyzer, or a digital oscilloscope could "tap" the internal processing done with the private key. In its preferred embodiment, the SCD performs no function other than the elementary private key computation needed in a particular application.

One of the two data communications connections, respectively the output connection for digital signatures, and the input connection for secret key establishment, is devoid of specific communications security precautions. This is an insecure logical connection, but since it is one-

way, there is no vulnerability to security breaches. The opposite data communications connection, respectively the input connection for digital signatures, and the output connection for secret key establishment, is logically connected to one or more secure "Functional application Area" (FA) using some security protocol, most likely an application-level protocol using long-term secret cryptographic keys as a basis for protection. An FA is a software-driven process usually implemented in a computer system, although it could be implemented in dedicated secure terminals. In the above digital signature example, the FA systems may be the computer systems that control the "final approval of digital signature release", and the application-level protocol may be a secret-key MAC operation on the data stream for which a digital signature operation is approved. Conversely, in the case of the PCT International application number 9852316A1, the FA system may be the computer systems which controls the actual verification of registrant's identity, and the application-level protocol may be a secret-key encryption operation (preferably preceded by some secret-key MAC operation).

The cryptographic keys shared between the SCD and the FA systems allow the distribution of SCD services (the elementary private key computation) to the FA systems that are secured to a lesser degree than the SCD. Thus, the present invention includes a system which protects the private key within the SCD while making the SCD services available to the FA systems. This is achieved through proper physical security protection, one-way data connections, and proper cryptographic key management facilities among other features disclosed in the detailed description.

The key management facilities of the present invention use the prior art technique of splitting a secret cryptographic key into a small number of key components for purposes of backup storage and transmission between secure devices. The preferred embodiment enhances this prior art technique by writing the key components in a specific digital media using a specific format so that the key components are not easily copied. One basis for this protection is the manufacturer's control over the design of the write and read interfaces for this specific media format. These interfaces are microprocessor-based and are required exclusively in devices that

are part of the present invention's product family. Thus, a "semi-proprietary" cipher algorithm may be programmed by the manufacturer into the interface microprocessors so that the format of the data is intelligible to read interfaces from other manufacturers for the same digital media (proprietary interface design). Clearly, for this to be perceived as an advantage, a user organization must trust the manufacturer's ability to maintain the secrecy of sensitive engineering details, which is a higher level of trust than the belief that the manufacturer's software does not leak key bits through a "subliminal channel" (this latter belief can be verified by an auditor). Another important basis for the fact that key components are not easily copied is the guaranteed unique (ROM based) serial number present in the specific media used in the present invention, namely the Dallas Semiconductor iButtons family of coin-shaped digital memories featuring memory sizes up to 64 kilo-bits. It is not easy to make a token that looks like a genuine Dallas Semiconductor iButton part but having some counterfeit electronics inside. In summary, the present invention combines cryptographic integrity and encryption algorithms with a proprietary interface design and unalterable serial numbers. The serial number is an input to some cryptographic integrity algorithm implemented in the read and write interfaces. This combination provides a media format which is difficult to copy. This deters legitimate users from planning fraud based on getting every component of a given key because such fraud would need to include skilled participants who might circumvent the copy protection scheme just described.

The implementation of one-way wired connections is unusual in the field of data communications. One approach is based on direct electrical connections over a very short distance that eliminates transmission errors for all practical purposes. Another approach is a standard protocol such as the HDLC LAP (High Level Data Link Control, Link Access Protocol) with a particular choice of options that prevents the data transmission in one direction. Negative acknowledgements or reject indications may nonetheless be transmitted to force retransmission upon error detection. In the preferred embodiment of the present invention, both approaches are used in a cascade arrangement: in each of the input and output connection, a network connection processor (NCP) lies between the innermost processor in the

SCD and the network. Both NCPs connect to the innermost processor using the short direct electrical connection approach and connect to other computer network elements using the one-way data link protocol approach. These other computer network elements can be a driver or a gateway that implements the particular protocol profile needed in the one-way data link protocol approach. Another possible approach for the implementation of the one-way data connection is the use of forward error-correcting codes, which are more commonly found in wireless links (e.g. satellite broadcast).

Brief Description of the Drawings

The figure 1 is a simplified illustration of the secure computing device and attachments in the cabinet enclosure according to the present invention.

The figure 2 is a block diagram for the printed circuit board in the secure computing device according to the present invention.

The figure 3 is a simplified electronic circuit diagram for the power supply monitoring and control by the secure computing device according to the present invention.

The figure 4 is a simplified circuit diagram for the program memory socket provided on the secure computing device according to the present invention.

The figure 5 is a simplified illustration of the key generation device according to the present invention.

Detailed Description of the Invention

The figure 1 gives an overview of the Secure Cryptographic Device (SCD) **101**, a main

component of the present invention, in its cabinet enclosure **100**.

The cabinet enclosure **100** is either a rack-mount or a desktop mount enclosure for electronics equipment, with physical security characteristics such as heavy duty steel panels, security hinges, lock and key, and restricted openings for heat dissipation vents and connections to the external power source and computer network connections. The cabinet enclosure **100** can be conventional equipment intended for a) computer room environment, b) harsh industrial environment, c) telecommunications industry, or d) self-service kiosk technology, provided that the physical access to the internal components shown in figure 1 is reasonably restricted to personnel in possession of the physical key for the cabinet.

The elementary private key computations are made by digital electronics located inside the Secure Computing Device (SCD) **101** which takes the form of a sealed Printed Circuit Board (PCB) with only those external connections and features that are essential to the main function of the SCD, namely the elementary private key computation. One type of digital integrated circuit that is readily available for the cost-effective implementation of the elementary private key computation is the Digital Signal Processor (DSP). In the preferred embodiment of the present invention, the SCD **101** encompasses an integrated cryptographic processor **102** (shown in figure 2) that takes the form of a DSP. A class of DSP well suited to the elementary private key computations is the integer DSP. The floating point DSP families typically dissipate relatively more heat for a given elementary private key computation. The Analog Devices' ADSP-21XX DSP microcomputer architecture has been found very well suited to the task of elementary private key computations when modular multiplications and modular exponentiations are the main compute-intensive operations. The Analog Devices' model ADSP-2181 offers a sufficient level of integration for a single-chip integrated cryptographic processor **102**.

The present invention isolates the private key in the cryptographic processor **102** with two separate data communication connections, respectively the one-way input serial port connection

105 and the one-way output serial port connection 106. Many models of DSP integrated circuits offer at least two serial ports in their system interface features. These serial ports can be used as such for the signal timing specifications for the input connection 105 and the output connection 106, while the electrical specification can be those of the receiver and driver circuits typically inserted behind input 105 and output 106 connections. These are short distance connections that eliminate transmission errors for all practical purposes. The input connection 105 (respectively output connection 106) is wired to the input Network Connection Processor (NCP) 107 (respectively output NCP 108). The primary role of the NCP's 107 and 108 is to convert the usual data communication protocols to and from conventions for the serial port connections 105 and 106. The isolation of the elementary private key computation from public networks is provided by the chain that links the input network connection point 109 to the input NCP 107, then to the input serial port connection 105, then to the integrated cryptographic processor 102, then to the output serial port connection 106, then to the output NCP 108, and finally to the output network connection point 110.

Specifically, the input serial port connection 105 can be made one-way by using only three signals out of the five present in the ADSP-21XX serial port, these signals being the serial clock (SCLK) signal, the receive frame synchronization (RFS) signal, and the serial data receive (DR) signal, all three driven by the input NCP 107. However, a flow control indication signal is required to enable the cryptographic processor 102 to indicate its readiness to accept the data for the next elementary private key computation. This is typically a single digital output signal driven by the cryptographic processor 102. By activating the three signals under its control, the input NCP 107 can feed the cryptographic processor 102 with the input data for the next elementary private key computation. With this arrangement, there is the smallest possible opportunity for the cryptographic processor 102 to transmit data to the input NCP 107, while providing a flow control signal to the input NCP 107.

The input NCP 107 may act as a standard network interface adapter that filters incoming data that is properly addressed to itself as a gateway to the cryptographic processor 102.

Alternatively, the input NCP **107** may be a special purpose data communications adapter that implements a standard protocol such as HDLC LAP (High Level Data Link Control, Link Access Protocol) with a particular choice of options that prevents the data transmission out of the NCP **107** through the input network connection point **109**. In this case, the input NCP **107** is physically and/or logically connected to another data network element (e.g. a driver or a gateway) that implements the particular protocol profile needed in the one-way protocol options approach.

A similar arrangement is used on the output side of the cryptographic processor **102**. For the output serial port connection **106**, the three serial port signals are the serial clock (SCLK) signal, the transmit frame synchronization (TFS) signal, and the serial data transmit (DT) signal. On this connection, these three signals are driven the cryptographic processor **102**. There is no important need for a flow control signal from the output NCP **108** to the cryptographic processor **102**. The output NCP **108** should have sufficient internal memory buffers to accommodate temporary network congestion that can occur on the output network connection point **110**. As with the input NCP **107**, the output NCP **108** can be either a standard network interface adapter that forwards the results of elementary private key computations through the output network connection point **110**, or a special purpose data communications adapter that implements the one-way protocol options approach.

Here is an example of one-way protocol options. In the standard HDLC specification, the optional functions 8 (respectively 9) limits the HDLC procedures to allow I frames to be commands (respectively responses) only. The HDLC I frame type is the one carrying payload information in the connection-oriented HDLC protocol. If the optional function 8 or 9 is used in a class of HDLC procedures using unbalanced operations (either the UNC or UAC class of procedure) and if the NCP has a fixed role either as an HDLC primary station or an HDLC secondary station, then the protocol specification is strictly speaking pure HDLC, but restricted to one-way data transmission. A representative option specification is if both the input NCP **107** and the output NCP **108** were HDLC secondary stations, respectively in UNC2,8 and UNC2,9.

According to "UNC2,8" for the input NCP **107** as an HDLC secondary station, the input NCP **107** would expect to receive command frame types I, RR, RNR, SNRM, and DISC, and would be allowed to send response frame types RR, RNR, UA, DM, FRMR, and REJ. According to "UNC2,9" for the output NCP **108** as an HDLC secondary station, the output NCP **108** would expect to receive command frame types RR, RNR, SNRM, DISC, and REJ, and would be allowed to send response frame types I, RR, RNR, UA, DM, and FRMR.

The NCP's **107** and **108** need not perform any specific cryptographic functions for the proper operation of the present invention. For digital signature applications, the "Functional application Area" (FA) systems are connected to the same network as the input network connection point **109**. The network connected to the output network connection point **110** should allow a logical connection with some computer application that distributes digital signatures according to the applications's needs (e.g. an X.500 directory service for X.509 security certificates). In a digital signature application, the one-way property of the output side of the cryptographic processor **102** is a security precaution against wrongful attempts to obtain a digital signature from the public network. In the case of secret key establishment applications, the input network connection point **109** is connected to the networks from which the individual requests for secret key establishment originate (e.g. a public network). Then, the one-way property of the input side of the cryptographic processor **102** is a security precaution against any possible leakage of cryptographic key material. The output network connection point **110** is logically connected to the FA systems that execute the secret key establishment processing downstream of the elementary private key computation (e.g. the queuing of applicant's electronic registration request for later verification of applicant's identity in the PCT International application number 9852316A1).

Another advantage offered by the present invention is a small deterrent against a number-theoretic attack to public key algorithms, namely the "chosen ciphertext attack". The chosen ciphertext attack has been studied in the academic literature, notably for "small exponent RSA"

and schemes based on the Rabin-Williams cryptosystem (where the public exponent is 2). To practically thwart the threat of "chosen ciphertext attack," it is advantageous to prevent an adversary from using the SCD at will (even without getting direct access to any secret key inside the device). This is achieved with the present invention because the input connection and the output connection to the SCD are linked to distinct computing environments.

In the preferred embodiment of the present invention, if the elementary private key computation is a digital signature then the final computation of the message digest for the digitally signed message is done within the cryptographic processor **102**. More than just the final computation can be done by the processor **102**, e.g. the whole message digest computation, but the key point is that a single one-way cryptographic function evaluation done within the processor **102** before a digital signature prevents the processor **102** from being "used against itself" in a chosen ciphertext attack scheme. For instance, if SHA-1 specification is used as the message digest algorithm, the processor **102** would be assigned the task of processing the final block of the padded message. More precisely, using the SHA-1 specification terminology, the cryptographic processor **102** would expect (as input data to the elementary private key computation received from input connection **105**) the final block of the padded message (the 16-word sequence are labeled W[0], W[1],..., W[15]), and the values stored in the 5-word buffer labeled H[0], H[1], H[2], H[3], H[4] as they appear just before the execution of the step b. in the last message block processing (the last message block is labelled M[n]). As an enhanced precaution, the processor **102** may be programmed to ignore input data that does not contain a properly padded last message block. The prior art teaches that the processor **102** would expect just the values stored in the 5-word buffer labeled H[0], H[1], H[2], H[3], H[4] after completion of final step, step e., in this said last message block processing, or that protection against the chosen ciphertext attack is enhanced if the processor **102** would be sent the complete digitally signed message. The present invention thus reduces the data transmission overhead to processor **102** while preventing the chosen ciphertext attack.

The programming of the cryptographic processor **102** implements the elementary private key

computation according to the specifications for a public key cryptography algorithm, an unambiguous data format syntax, and some network addressing conventions, as the case may be for forwarding the elementary private key computation results to the proper destination. The present invention allows flexibility in the cryptographic processor **102** programming through the program memory socket **120** while ensuring stored program integrity through the 8-segment numeric display **121**. The program memory socket **120** is a simple standard socket for memory integrated circuit, or an equivalent interface for some type of program storage memory (EPROM, Flash, serial EPROM, and the like).

Two mechanisms are provided in order to prevent bogus software from being run by the cryptographic processor **102** in spite of the programming flexibility allowed by the program memory socket **120**. The first such mechanism is the 8-segment numeric display **121** (e.g. pocket calculator type LCD) that displays some conventional hash code for the software that is read from the program memory socket **120**. This conventional hash code can be the SHA-1 and the 8-segment numeric display **121** can display, in a rotating sequence, 8 hexadecimal digits at a time (e.g for a duration of 5 seconds each) out of the 40 digits SHA-1 hash code, followed with a blank display (e.g for a duration of 2 seconds). Thus, through a display interface with simple electronics, an operator looking at an SCD **101** can tell for sure that the software loaded in the cryptographic processor **102** is a specific one. In practice, the software integrity is ensured if, as part of a software release procedure, the developer certifies (e.g. on a paper certificate) the correct SHA-1 code value (to be displayed once the officially released software is loaded), and a security auditor of the SCD operation looks at the paper certificate before checking the SHA-1 actually value displayed by the 8-segment numeric display **121** during a visit to the SCD premises. The other mechanism that prevents bogus software loading is a mechanical protection (e.g. a physical hinge, a small door, a lock and a key) that controls access to the program load area **104** where the program memory socket **120** is located. Thus, introducing a new version for the software run by the cryptographic processor **102** is a restricted operation.

The operating mode of the SCD **101** is controlled with the 4-position switch **111**. When the

switch **111** is in the "Program load" position, all cryptographic key material inside the SCD **101** is erased and the cryptographic processor **102** software is loaded from the program memory socket **120**. When the switch **111** is in the "Key setup" position, the key component interface **119** and the random digits keypad **118** are enabled.

Many cryptographic schemes require a source a random secret bits and a conventional technique to provide them is a pseudo-random number generator (PRNG) inside a physically secure computing device such as the SCD **101**. The presence of this requirement depends on the cryptographic algorithms and protocols implemented by the stored program (e.g. many discrete logarithm based public key algorithms require random secrets generation, as do most session key establishment protocols). As is well known in the field, the PRNG approach can be fast and reliable provided that the state information is properly initialized and is always kept secret. Then, upon initialization, the internal state information of the PRNG must be set to totally random values that virtually nobody should have the slightest clue. For this purpose, when the switch **111** is in the "Key setup" position, the random digits keypad **118** is available for any operator to enter arbitrary digits that the operator should forget. Another source of random data is the sampling of a timer upon keystroke interrupts.

The key component interface **119** is used to insert the internal configuration into the cryptographic processor **102**. This configuration includes notably cryptographic key material according to a key management scheme such as the one described below. Through the key component interface **119**, the operator can inject "configuration control records" into the SCD **101**. The key component interface **119** has to be compatible with the media on which the configuration control records are written. Some mechanical protection (e.g. a physical hinge, a small door, a lock and a key) controls access to the key management input area **103** where the key component interface **119** is located. Thus, injecting configuration control records into the cryptographic processor **102** is a restricted operation.

In the preferred embodiment of the present invention, this media is iButton memory tokens

(e.g. part numbers DS1992, DS1993, DS1995, or DS1996 in NVRAM technology, part numbers DS1982, DS1985, or DS1986 in EPROM technology) using some proprietary obfuscation scheme as a first line of defense for cryptographic key material confidentiality. Then the key component interface **119** is a very simple receptacle (e.g. part numbers DS9098 or DS9100) with a "one-wire bus" electrical interface easily interfaced to a microcontroller with adequate electrical protection. In addition, the firmware makes the key component interface **119** read-only, to prevent any leakage of cryptographic key material. Other combinations of media and interface are possible for respectively the configuration control records and the key component interface **119**. The key component interface **119** can be a bar code reader if the media is a bar code printout. To enhance the security in the case of bar code media, a "black-on-black" scheme might be used where a special purpose bar code reader can distinguish light intensity variations outside of the visible spectrum (infrared or ultraviolet), and where a printer equipped with two special ink reservoirs and jets, respectively for true black and visible-lightwaves only black. These bar codes are concealed in a seemingly plain black area of a printout and are difficult to copy with ordinary photocopying equipment. Another alternative for the key component interface **119** is an ordinary keyboard, where the media is a human readable printout, but this alternative is not very convenient given the relatively large size of the cryptographic key material for public key cryptography. Yet another alternative would be a paper punch reader for the key component interface **119** with the corresponding media being old-fashioned strips of paper punch.

Unless the SCD 4-position switch **111** is in the "Operating" position, the SCD **101** does not accept any input data through the input serial port **105**. The SCD 4-position switch **111** "Standby" position stops the normal operation of the public key application: it prevents the SCD **101** from accepting any input data through the input serial port **105** even if the SCD **101** configuration is complete.

The figure 2 shows the arrangement of the printed circuit board inside the SCD **101**. This

arrangement preserves the security of the private key. The integrated cryptographic processor **102** being a modern digital signal processor (DSP) integrated circuit, it can integrate all the required digital electronics in a "secure chip" arrangement: a CPU (central processing unit) notably well suited to the public key computations, sufficient internal (on-chip) memory for program storage, configuration data, buffers for input serial port connection **105** and output serial port connection **106**, and any necessary intermediate results in the elementary private key computation. Many modern DSP integrated circuits offer a low power mode in which the program logic can enter and during which the DSP integrated circuit operates at a reduced clock frequency and retains all its internal memory contents while taking a fraction of the normal operating current. This allows the external battery **124** to supply sufficient power for the expected maximum duration of a power outage on the main power connection **122**.

In addition, many modern DSP integrated circuits are designed with an external memory bus that can be shared in a multi-processor configuration. Typically, this feature is implemented with a control signal (e.g. "bus request") that puts the DSP external memory bus in a tri-state mode, which enhances the electrical isolation of the memory inside the cryptographic processor **102** from the external world. Then, the only output signals from the cryptographic processor **102** are the signals for the output serial port connection **106** and the flow control signal to the input serial port connection **105**. In the preferred embodiment of the present invention, the control signals **130** includes the following ADSP-2181 input signals: a) the reset (RESET) signal, b) the Bus Request (BR) signal, and c) the powerdown (PWD) signal. The inputs to the integrated cryptographic processor **102** are a) control signals **130** from the power, reset, and tamper-protection control circuit **112**, b) the internal memory initial program load interface **128**, c) the input data port **129**, and d) the input serial port connection **105**. Each of these four interfaces has an independent logical function and is electrically connected to a different section of the printed circuit board.

A possible arrangement is to replace the flow control signal by a "device operating properly" signal **132** driven by the power, reset, and tamper-protection control circuit **112**, as an indication

that the SCD 4-position switch **111** is in the "Operating" position and that all other conditions for normal operation are met. This alternative arrangement can be considered more secure because the integrated cryptographic processor **102** has virtually no way to leak key material through the input serial port connection **105**, and it can be practical if the input NCP **107** is programmed with some timing logic and configured with an a-priori knowledge of the rate at which the cryptographic processor **102** can process input data (elementary private key computation latency).

In the case of absence of main power (normal power connector **122**), the external battery connection **123** is used to maintain the integrated cryptographic processor **102** in a low power mode where the memory contents is preserved but otherwise the processing capability of the cryptographic processor **102** is frozen. This condition is visually indicated to the user because the LED indicator **113** labelled "Battery" is lit while the LED indicator **113** labelled "Power" is off. For normal daily operations, this condition is further notified to the user because the LED indicator **113** labelled "Standby" becomes sticky, i.e. remaining lit if the switch **111** is in the "Standby" position or in the "Operating" position. For maintenance operations, this condition is also notified to the user because the two LED indicators **113** labelled "Program Load" and "Key Setup" remain off even if the switch **111** is in the corresponding position. In all cases, bringing the 4-position switch **111** to the "Program Load" position erases the memory contents in the integrated cryptographic processor **102**. The purpose of the internal battery **125** is to supply the energy needed for the erasure operation upon detection of any condition that requires it. Upon detection of a condition that triggers the erasure of internal memory in integrated cryptographic processor **102**, the internal battery **125** may need to supply power for the time needed for the erasure operation to complete.

The program load and memory erase processor **114** controls the loading of the processor **102** internal memory through the internal memory initial program load interface **128**. In the preferred embodiment of the present invention, the ADSP-2181 internal direct memory access

(IDMA) port is used for boot loading the integrated cryptographic processor **102**. Again, to maintain the isolation of the internal memory in processor **102** from the outside world, the ADSP-2181 IDMA port read strobe signal (IRD) is held high so that the IDMA port can not be used to read the processor **102** internal memory. In addition, the IDMA port write strobe signal (IWR) is gated by a signal that is part of the control signals **115** and driven by the power, reset, and tamper-protection control circuit **112**. This way, the cryptographic processor **102** is protected by the control circuit **112** from a faulty program logic in the program load and memory erase processor **114** during normal operation. It is clear for someone familiar with the field of DSP programming how the implementation details for the program memory socket **120** can be used by a software developer, along with DSP software development tools, to program the cryptographic processor **102** according to the detailed specification for the load format recognized by the processor **114**.

The role of the program load and memory erase processor **114** is: 1) to erase the entire cryptographic processor **102** internal memory (through control of the internal memory initial program load interface **128**) whenever the 4-position switch **111** is brought to the "program load" position, or upon detection of any condition that requires such memory erasure by the control circuit **112**, as indicated through the control signals **115**, 2) to read the memory contents from the device inserted in the program memory socket **120**, whenever the 4-position switch **111** is brought out of the "program load" position, and other preconditions for normal operation of the present invention are encountered by the control circuit **112**, as indicated through the control signals **115**, 3) while reading the memory contents from the socket **120**, loading the integrated cryptographic processor **102** through the internal memory initial program load interface **128**, and computing the conventional hash code (e.g. SHA-1) of this said memory contents, and 4) on a periodic basis (e.g. every 42 seconds or so) (while the 4-position switch **111** is in a position other than "program load," and while the control circuit **112** detects a normal power supply condition and no condition that requires erasure of processor **102** internal memory) reading the memory contents from the device inserted in the program memory socket

120, and computing the conventional hash code (e.g. SHA-1) of this said memory contents, and finally comparing the just computed hash code with the hash code that was initially computed when the 4-position switch 111 was brought out of the "program load" position (as a conclusion of this periodic computation of the hash code value, the processor 114 should refresh the data displayed on the display 121 and trigger an alert control signal 131 upon detection of any anomaly such as hash code mismatch or the absence of a memory device in the socket 120). The program load and memory erase processor 114 is key-less, that is, it computes a conventional hash code and displays it on the operator display 121 but does not prevent the SCD 101 from operating normally based on the actual value of the initially computed hash code value. The alert control signal 131 is handled by the power, reset, and tamper-protection control circuit 112 as other tamper-detection signals.

The connections to the key import processor 116 are closely related to the functional logic to be programmed in this processor 116. Through the control signals 117, the key import processor 116 will be started whenever the 4-position switch 111 is in the "Key Setup" position. Then, the role of the key import processor 116 is to detect the presence of iButton memory tokens on the key component interface 119, read their contents, then apply the proprietary obfuscation scheme to de-"obfuscate" any obfuscated portion of a part of a configuration control record. Once so de-"obfuscated", the part of a configuration control record just read is immediately fed into the integrated cryptographic processor 102 and then erased from the processor 116 memory. The logic required for the combination of various parts of a configuration control record, once de-"obfuscated", lies within the cryptographic processor 102 so that sensitive cryptographic key material is never present in cleartext form within key import processor 116. Through the random digits keypad 118, the key import processor 116 receives only one type of information: random key strokes. The actual values of the keys that are entered and the timing information of strokes is also fed to the integrated cryptographic processor 102 with an indication of the type of data being fed. The logic that turns this data into a random secret PRNG state information lies within the processor 102. Realistically, the processor 102 may

refrain from operating normally until sufficient random key strokes are entered. Among the control signals **130** to the processor **102**, there is an indication that the 4-position switch **111** is in the "Key Setup" position. Under this condition, the program logic within the processor **102** can inter into a poll loop for data to be received from the input data port **129**. With the ADSP-2181 found in the preferred embodiment of the present invention, the poll loop may read an I/O register location in the I/O memory space, this register being filled by the key import processor **116** whenever something has to be transmitted through port **129**.

The figure 3 shows the detailed arrangement of a section of the control circuit **112** in the case of the preferred embodiment. The circuit section shown on figure 3 is based on the prior art electronic techniques for supplying and monitoring a direct current (DC) source for microcontroller based printed circuit board. The circuit section shown on figure 3 addresses the need to maintain +5 Volts DC supply from the external battery **124** (in the case of external power **122** outage) for the preservation of cryptographic key material inside the cryptographic processor **102** and +5 Volts DC supply from the internal battery **125** for the very short period during which the program load and memory erase processor **114** erases this same cryptographic key material. The DC/DC voltage regulators **133** and **134** are controlled (turned on and off) by the microcontroller inside the control circuit **112** through two independent digital output signals (labelled respectively OUT_X and OUT_Y in the figure 3). During normal operation, all four voltage monitors shown in figure 3 indicate a normal voltage condition to the microcontroller input signals labelled RESET, INT_X, NMI, and INT_Z and the DC/DC voltage regulators **133** and **134** are turned off to spare the energy in batteries **124** and **125** respectively. At the instant when the external power **122** is cut, the voltage monitor connected to the microcontroller INT_X input signal triggers an interrupt (or the voltage drop is otherwise detected, e.g. in a programmed polling loop inside the microcontroller logic). The DC/DC voltage regulator **135** and/or the capacitors coupled to the +5 Volts supply plane in the printed circuit board provide enough energy for the microcontroller to quickly react to the power outage by turning on the DC/DC voltage regulator **133** fed by the external battery **124**.

The orderly shutdown of the SCD **101** is under the supervision of the microcontroller inside the control circuit **112**. The orderly shutdown is triggered if the external battery power **123** are lost. This orderly shutdown is achieved by the microcontroller by 1) quickly turning on the DC/DC voltage regulator **134**, 2) commanding the program load and memory erase processor **114** to erase all of the cryptographic processor **102** internal memory, 3) upon receiving a confirmation through control signals **131** that such erasure is complete, turning off first the DC/DC voltage regulator **133**, then the DC/DC voltage regulator **134**. In this sequence, the internal battery **125** is used for the time it takes to erase the processor **102** internal memory. At the end of this sequence, the microcontroller inside the control circuit **112** effectively cuts its own power supply. Then, the voltage monitor connected to the microcontroller RESET signal puts the microcontroller in the RESET state. The normal procedure for the SCD **101** to start from a no-power condition is to insert good batteries as the external battery **124**, and then to apply the proper DC Voltage to the external power connection **122**. At this point, the voltage monitor connected to the microcontroller RESET signal will put the microcontroller out of its RESET state and the microcontroller will start monitoring its various inputs, notably any change in the position of the 4-position switch **111**.

It will be understood by someone skilled in the art of microcontroller-based systems how the program logic for the microcontroller inside the control circuit **112** can be implemented to account for the various conditions that may arise and command the operating state of cryptographic processor **102**, key import processor **116**, and program load and memory erase processor **114** according to the rules set forth in the present invention disclosure, and to provide operator feedback through the LED indicators **113**. For instance, it is obvious for someone skilled in the art of microcontroller-based systems that the microcontroller has to debounce the 4-position switch **111**. Overall, the figure 3 shows a circuit diagram that gives full control to the program logic inside the microcontroller in the control circuit **112** over the use of the external battery **124** and the internal battery **125**. Some key points are to provide to the microcontroller some input signals that reflect the presence of valid input voltage for every individual power source that the SCD **101** may use for its operation, and to make sure that the electrical

characteristics of the voltage regulation and capacitors do supply operating voltage for the sub-millisecond delay that the microcontroller needs to turn on the alternate voltage supply. Then, it will be understood why the microcontroller NMI signal is connected to the external battery connection **123**, that is because NMI stands for "Non-Maskable Interrupt" in microcontroller terminology and a non-maskable interrupt means a very fast response time from the microcontroller who must to turn on the internal battery **125** as fast as possible.

The figure 4 shows a diagram of the circuit section in the SCD **101** that ensures proper electrical isolation of the program memory socket **120** from the rest of the circuit. A pulse-width modulation (PWM) (e.g. frequency greater that 100 MHz) drives an isolation transformer so that the DC power applied to the socket **120** is isolated from the main power supply for the rest of the printed circuit board. A control signal from the circuit **112** drives the PWM circuit so that the presence of a valid voltage on the interface **120** is controlled by the control circuit **112**. The power, reset, and tamper-protection control circuit **112** is responsible for turning off the PWM circuit for the duration that the switch **111** is left in the "program load" position, so that the program memory socket **120** is not "live" when the operator is allowed to replace the memory device inserted in the socket **120**. Optocoupling circuits are provided between the address, timing, and data signals present in the socket **120** and the corresponding interface pins for the program load and memory erase processor **114**. Similar arrangement can be made for the key component interface **119**. The figure 4 shows an 8-pin interface in the socket **120**, as is typical for a serial programmable read-only memory (SPROM) sold for digital electronic products that include field-programmable gate arrays (FPGA). The capacity of currently available SPROM integrated circuits matches the 640 Kilo-bit total internal memory size found in the ADSP-2181 processor used as the processor **102** in the preferred embodiment of the present invention.

Known rules of the art for electronic circuit design and manufacture should be followed to minimize the electromagnetic emissions from the SCD **101**, and also to protect the internal circuits from high voltages from any externally accessible electrical connection. Two tamper

detection loops are shown on figure 2, respectively the normally open tamper detection loop **126** and the normally closed tamper detection loop **127**. These circuits are provided for the control circuit **112** to monitor the physical integrity of the envelope surrounding the SCD **101**, and to force the program load and memory erase processor **114** to erase all of the processor **102** internal memory upon detection of either the shorting of the normally open loop **126** or the breaking of the normally closed loop **127**.

The "Key Generation Device" (KGD) is a special purpose electronic device that is capable of generating pairs of public and private keys, and which has peripherals for securely writing media that holds cryptographic key material. The role of the KGD could be performed by the SCD, but it requires more user interaction and input/output activities than what should be allowed from a dedicated high security public key cryptography processor like the SCD. The difference between the KGD and the SCD in this respect is that the KGD memory is devoid of any cryptographic key material when not in use and under the direct supervision of security personnel when in use. The SCD contains critical cryptographic keys and is under the control of operations personnel most of the time, so any user output channel theoretically represents a potential path for fraudulent leakage of key material.

The figure 5 shows a possible configuration for the key generation device (KGD). The KGD is a system used occasionally for key management operations performed by trusted personnel. It is made of three components: a stand-alone server cryptographic device (SCD) **101**, a personal computer **141** with a key management supervisory software program made for the purposes of the present invention, and a KGD peripheral controller **136**. Basically, the KGD design takes the first two of these items (SCD **101** and personal computer **141**) because they are available anyway, and puts anything that is missing on the third one (peripheral controller **136**). The KGD peripheral controller **136** is a single board computer with the peripherals and connections required to fulfill its specific roles.

A function assigned solely to the KGD peripheral controller **136** is the writing of media used

for the configuration control records. While the key component interface **119** on the SCD **101** is read-only, the KGD peripheral controller **136** is equipped with a key component write interface **137** that is write-only. To account for the iButton devices using the EPROM technology, the KGD needs a modified one-wire bus electrical interface to supply the programming voltage typical to the EPROM circuits. The proprietary obfuscation mechanism used in the present invention is implemented by the microcontroller that drives the key component write interface **137**. This obfuscation mechanism requires a source of random secret bits and for security reasons, the PRNG within the SCD **101** can not be used for this purpose as a generic source of random secret bits (otherwise, the PRNG output would be exposed through the interface **106**, which would open the door to cryptanalysis attempts). Thus, the KGD peripheral controller is equipped with a keypad **139** used notably for the input of PRNG state information as described previously for the keypad **118**.

In the context of the present invention, "technology obsolescence" refers to the case where some portion of the system is no longer available and it is important to implement the logical operations of the system (its programming) in a new computing environment. Then, the public/private key pair (PPKP) value must be recoverable without recourse to any proprietary media, interfaces or algorithms. Then, the PPKP may be safeguarded in multiple components using "technology obsolescence recovery media," that is simply a printout using the paper slip printer **138**. Preferably, the paper slip printer should be devoid of any buffer memory that can not be reliably erased.

The security and trust model behind this KGD design is that the PPKP generation procedure is a very critical event in the life-cycle of a private key, yet it makes business sense to protect it by trusted personnel and a KGD system designed with less absolute security requirements than in the case of the SCD **101** in its enclosure **100** used in a production environment. An example of a cost element that can be avoided in the KGD design by this trust model is a secure tamper-proof enclosure that would hide the output serial port connection **106**. In the present invention, the program logic and the memory buffers inside the KGD peripheral controller **136**

are a potential source of security weaknesses. Accordingly, the firmware inside the KGD peripheral controller **136** should be as simple as possible, in order to facilitate its auditing. No multi-tasking kernel should be included. The programming model should be limited to a simple command loop with three input queues respectively for data received through the serial port **106**, the data received from the personal computer **141**, and the keystrokes from the keypad **139**, four blocking write primitives respectively to the key component write interface **137**, the paper slip printer **138**, the operator display **140**, and the personal computer **141**, and finally a blocking read primitive (with a time-out) for the KGD key component read interface **143**. To assist firmware integrity auditing, the hash value of the firmware (e.g. SHA-1 value) can be displayed on the operator display **140** upon firmware boot.

One primary function of the KGD is the generation of the public/private key pair (PPKP), of which the private counterpart is the single most important secret. The computations required for the generation of a PPKP are dependent on the public key algorithm in use. For discrete logarithm cryptosystems, the generation of a PPKP is typically trivial, that is the selection of a secret random number, and then the inclusion of a complete SCD **101** in the KGD design is an overkill. For cryptosystems based on the difficulty of factoring large integers, the generation of a PPKP encompasses the generation of very large random prime numbers, a task that requires a reliable source of secret random bits, ample processing power, and nearly identical secrecy protection as for the private key usage. The processing power required with the ADSP-21XX DSP microcontroller family has been found to be in the order of hours or days depending of the key sizes and the number-theoretic constraints applied to the prime numbers allowed by the generation process. Since the generation of a PPKP is very infrequent, this level of performance is adequate.

Important key management attributes must be affixed to the PPKP to ensure usage integrity for the private key just generated. One such attribute is an arbitrary number (e.g. 32 bits) as the (non-secret) unique identification "PubK" of the PPKP to be generated, the scope of this identifier being the key management operations for a given organization. These PPKP attributes

also include an object identifier "PubKObjId" (global scope), some top-level key exchange key, and a couple of flags. It is now possible to describe the process by which the KGD generates a PPKP. The following procedure is applied:

1 The generation process is triggered by the insertion of a PPKP generation program in the program memory socket **120** and the manual movement of the 4-position switch **111** out of the "Program Load" position.

2 The PPKP key management attributes are presented to the KGD peripheral controller **136**. The PPKP key management attributes are input from the keypad **139** or through the connection with the personal computer **141**, except for the key exchange key which is generated locally by the KGD peripheral controller **136**.

3 When the 4-position switch **111** is brought to the "Operating" position, the actual PPKP generation starts.

4 The PPKP generation program first sends through the output serial port connection **106** a secret random bit string of sufficient length as a keystream for a one-time pad cipher used later for the transmission of the PPKP to be generated (another way to put it is that a first "raw" key component for the PPKP is sent through connection **106**).

5 Once informed of the PPKP generation program beginning and having received the PPKP key management attributes, the KGD peripheral controller **136** is in a position to write all but one of the parts of a configuration control record planned for the PPKP. The number of such parts may be entered on the keypad **139**. Regular key components in the form of parts of a configuration control record are written through the key component write interface **137**, and any technology obsolescence recovery media are printed on the paper slip printer **138**.

6 The PPKP generation program should have a minimal delay of e.g. 1 hour before completing its task to make it possible for every parts of a configuration control record written so far to be safely put aside before the second component of PPKP is sent.

7 Upon completion of the PPKP generation process, the SCD **101**, transmits to the KGD

peripheral controller **136** the complete PPKP encrypted with the one-time-pad keystream sent previously (another way to put it is that the second "raw" key component for the PPKP is sent through connection **106**). At this point, the KGD peripheral controller **136** is in a position to write the part of a configuration control record for the PPKP (again in dual media, respectively regular key component and technology obsolescence recovery media).

At the end of this process, the KGD peripheral controller **136** must erase all traces of the PPKP in its memory buffers. A reset pushbutton **142** is provided on the KGD peripheral controller **136** to ensure that this is done (within the firmware boot logic). The PPKP generation program inside the SCD **101** should do the same, but in this case, a manual movement of the 4-position switch **111** to the "program load" position will achieve this same goal irrespective of the logic in the PPKP generation program.

Variations of this process are possible, but it is difficult to prevent security consequences of any eavesdropping on the connection **106** without encumbering the key management scheme with e.g. an additional kind of key exchange key. The PPKP key management attributes could be affixed to the PPKP by the SCD **101**, but this would add some complexity in the PPKP generation program without simplifying the KGD peripheral processor **136** firmware logic.

Once generated and stored on a number of parts of a configuration control record (each part holding a key component obfuscated according to a semi-proprietary scheme), the PPKP is readily conveyed to a "Server-side Public Key Cryptography Apparatus" (SPKCA). We use the term SPKCA for an SCD **101** installed in a cabinet enclosure **100** and used in a production environment, in order to distinguish it from an SCD **101** connected to a KGD peripheral controller **136**. From the knowledge of the PPKP, the SPKCA is able to compute the public key and perform the elementary private key computation. As explained in details below, the SPKCA spontaneously produces "hello messages" that announce the public key(s) for which the SPKCA is enabled. This process might be important when the public key algorithm is based on the discrete logarithm and the KGD peripheral controller **136** is able to select a private key but

unable to compute the public key counterpart (e.g. simply because no large integer arithmetic library is readily available for the processor used in the controller **136**). Thus, it is feasible to avoid the cost of an SCD **101** in a KGD design for a discrete logarithm cryptosystem.

For public key algorithms based on the difficulty of factoring large integers, for the very involved PPKP generation procedures (such as those discussed in the above mentioned Ueli Maurer's article), the processing capabilities of the SCD **101** may not be sufficient. Then, the PPKP generation procedure described above can be adapted using the personal computer **141** for the operations done by the SCD **101**. This approach is potentially insecure because the personal computer is an insecure environment, but it avoids the cost of an SCD **101** in a KGD design. It is indeed the approach taken by the USA patent 5,675,649 for key generation (although without any semi-proprietary obfuscation mechanism).

The ultimate function of the key management for the SPKCA and KGD is that they together provide each legitimate "Functional application Area" (FA) with a "Key Exchange Key" (KEK_{FA}) that enables routine transactions between the FA and the SPKCA, while maintaining the highest standards of protection for the critical cryptographic keys used in the SCD and manipulated by the KGD. In the course of key management operations, the KGD must read the contents of configuration control records. The KGD key component read interface **143** is provided for this purpose. Two independent interfaces, respectively for reading **143** and writing **137** parts of configuration control records, are provided in order to reduce the risk of operator errors when manipulating the iButton tokens.

A FA can be any computer application that performs a processing function downstream (respectively upstream) of the public key decryption step (respectively the public key signature step) performed by the SPKCA. An example is the secure computer system of a "local registration agency" when the SPKCA is used as a digital signature device for a certification authority. Typically, the user access to a FA is controlled by passwords and/or other authentication mechanisms, but the FA security level might be somewhat lower than the

security level intrinsic to the SPKCA and KGD design. Accordingly, a key hierarchy is provided so that the more likely security breaches in a FA can be fixed by relatively more frequent key changes. The key at the top of the key hierarchy is the private counterpart of a public key which is likely to be broadly distributed. In practice, a key change for the key at the top of the key hierarchy should be avoided at all cost. The cryptographic key hierarchy inside the SPKCA is shown in table 1.

In its simplest expression, the key management scheme comprises

1 a PPKP pair of public and private keys of which the private key counterpart is never stored outside of one or more SPKCAs except as provided by the detailed key management scheme described herein,

2 a KEK_{BK} and a KBK which are directly linked to the PPKP in a given SPKCA,

3 a set of KEK_{FA}'s, where each one links a given functional application area to one or more SPKCAs that share the same PPKP.

The application link to the cryptographic services enabled by the PPKP private key counterpart is protected by the KEK_{FA}. The KEK_{FA} is injected in a SPKCA using its current KBK, and the KBK is renewed using the KEK_{BK}.

As a security precaution, even if a key KEK_{FA} might be shared between more than one SPKCA, it should not be shared between FA computer systems. Otherwise, in a secret key establishment application, a defrauder could use a stolen FA system to recover the KEK_{FA}, and then impersonate the function of a SPKCA.

Short Name	Full Name	Description	Shared Among	Backup Storage	Record Types
PPKP	Public/Private Key Pair	A key pair according to some public key cryptography primitive (digital signature or secret key establishment)	Used locally by the SPKCA (only the public key is shared)	Corporate safe boxes	SPKCA: 'A'
KEK_{BK}	Key Exchange Key for KBK	A secret KEK that isolates lower keys from the private part of PPKP	The SPKCA and the KGD	Corporate safe boxes	SPKCA: 'B' KGD: 'b'
KBK	Key Binding Key	A secret key that controls the bound between PPKP and KEK_{FA}.	The SPKCA and the KGD	Security group safe boxes	SPKCA: 'C' KGD: 'c'
KEK_{FA}	Key Exchange Key for KFA	A secret KEK that secures the application-specific association between PPKP and a Functional Area	The SPKCA, some FA system, and the KGD (each KEK_{FA} is shared with a FA system)	The KEK_{FA} secure database should be backed up	SPKCA: 'E', 'F', 'G'
KFA	Key for Functional Area	A secret working key used in a limited number of transactions	The SPKCA and some FA system	n/a	
LMK	Local Master Key	A local master key that protects the KEK_{FA} secure database	Used locally by the KGD	Security group safe boxes	KGD: 'D'

Table 1) Cryptographic key hierarchy inside the SPKCA and the KGD

The complete picture for key management most likely includes, in addition to SPKCA, KGD, and FAs, some safe boxes for storage of backup cryptographic key material. Usually, a minimum of two safe boxes are used to store a backed up key, each controlled by a different custody. Since the key at the top of the key hierarchy must be strongly protected, it is recommended to isolate the backup of this particular key (the private key) in separate safe boxes from the secure storage used by the information security personnel for less critical backup cryptographic key material. We use respectively the terms "corporate safe boxes" and "security group safe boxes" for the most secure and the relatively less secure environments. For instance, two corporate safe boxes might be under the responsibility of respectively the legal affairs department and the external auditors, while the information security personnel might report to the chief financial officer and use the treasury's own safe storage as the security group safe boxes. With such an arrangement, a suspected collusion between individuals among the information security personnel might be recovered without having to change the widely distributed public key. Here is why: an adequate recovery from a suspected collusion must include the re-keying of all cryptographic systems and applications under the control of the impostors, but since the backup private key was under independent control, it is a reliable starting point for the re-keying operation. Taking a certification authority digital signature key as an example, the external impact of the recovery might be limited to revocation and re-issuance of certificates issued during the period of suspected collusion. Although this is a serious impact, it is less dramatic than the adoption of a new private key, the distribution of a the corresponding new public key, and the re-issuance of every certificate because the former ones were voided by a collusion among employees having access to safe boxes where the former private key backups were kept.

The management of the long term application keys KEK_{FA} is preferably done with a list management function built-in the KGD. Each entry in this list would include

1 a reference "PubK" to the PPKP public key counterpart,

2 a reference "FAid" to the functional application area,

3 a flag indicating the presence of a key KEK_{FA} for this FAid,

4 a cryptogram of this KEK_{FA} if present, and

5 maybe a check digit for this KEK_{FA} if present.

As is well known from the prior art, the whole list could be stored in a secure file protected by a "local master key" (LMK) in a storage media accessible to the KGD during the key management user session. The secure operations on the list elements would include

- creation of a new entry (either random key generation by the KGD or key import through typically two or more components) (record type 'F'),
- renewal of the KEK_{FA} (either random key generation by the KGD or key import through typically two or more components) (record type 'F'),
- wiping out of KEK_{FA} while retaining the FAid (the erasure of KEK_{FA} has to be transferred to every SPKCA before the FAid could be deleted) (record type 'G'),
- wiping out every KEK_{FA} at once (record type 'E'), and
- entry selection for updating the status of a related SPKCA (the process enabled by the current KBK in the SPKCA) (from record type 'c').

To illustrate the intended process, a typical key management user session is a step-wise process described below:

1 the KGD gets hold of the LMK if necessary (from record type 'D'),

2 the user access rights are validated (if not done as a part of the previous step),

3 the user may request the (indirect) erasure of every KEK_{FA} in remote SPKCAs (record type 'E'),

4 the user manages the KEK_{FA} list: he may insert new keys (record type 'F'), renew some other keys (record type 'F'), and wipe out existing keys (record type 'G') as the case may be (an attractive user dialogue feature for this step is automatic selection of changed entries),

5 out of the KEK_{FA} list, the user selects entries that he wishes to synchronize in the current configuration of one of more SPKCAs (e.g. the entries that were changed in the previous step for an SPKCA configuration update, or every entries for the configuration of a new SPKCA or after having requested erasure of every KEK_{FA}),

6 for each SPKCA in which the configuration should be (indirectly) updated, the current key KBK is presented to the KGD (from record type 'c'),

7 the KGD produces the required encrypted configuration control records for indirect configuration update for the selected entries (record types 'E', 'F', 'G'),

8 the updated KEK_{FA} list is written back to the storage media, encrypted under the LMK,

9 the user authorization status is erased from the KGD,

10 the LMK is erased from the KGD if it was not present at the outset of this process.

For the purpose of managing the list of KEKFA, the KGD peripheral controller **136** is used as a general purpose cryptographic key management device. The keys KEK_{FA} are conventional secret keys which can be input to KGD peripheral controller **136** through the keypad **139** in multiple components, and output from the KGD peripheral controller **136** either through the operator display **140** or the paper slip printer **138**. The database of KEK_{FA} is encrypted under the LMK and can be stored on the hard disk present in the personal computer **141**. Any manipulation of a cryptographic key in the clear can be implemented in the program logic in the KGD peripheral controller **136** firmware.

The configuration control records are structured messages that are intended to the SPKCA or the KGD for conveyance of key management information. Each record may be broken in a number of parts, and a record may not be decrypted by the SPKCA or KGD before all of its parts are presented consecutively. In addition, the contents of a configuration control record may, depending on its type, be protected by an independent secret key (using the prior art notion of "key exchange key"). Each part of a configuration control record is structured according to the following model:

- a media serial number,
- a cleartext record type tag, made of three characters indicating respectively the type of configuration control record, the "part number" digit, and the "part count" digit (e.g. "A13" represents the part 1 of 3 parts of a configuration control record of type "A"),
- an optional cleartext indication, "PubK", of the public key to which the record applies,
- an arbitrary record identifier (e.g. a 32 bits random number), "RecId", common to every parts of the configuration control record,
- an "obfuscated" secret key component, where "obfuscated" refers to a semi-proprietary cryptographic scheme described below,
- (only in the case of the first part) a payload cryptogram, protected by a key that is the "exclusive or" of (de-"obfuscated") secret key components of every parts of the configuration control record, "exclusive or-ed" again, as the case may be, with the associated independent secret key ("key exchange key").

Here is a representative example for a configuration control record that has no associated "key exchange key":

- SerialNo¦'A1#'¦PubK¦RecId¦Obfusc(KA1)¦$PROT_{KA1 \oplus KA2 \oplus \ldots \oplus KA\#}$('A#'¦PubK¦RecId¦PubK ObjId¦PPKP¦KEK_{BK}¦Flag_B¦Flag_C), as the first part for the private key,
- SerialNo¦'A2#'¦PubK¦RecId¦Obfusc(KA2), as the second part for the private key, if any,
- ...
- SerialNo¦'A##'¦PubK¦RecId¦Obfusc(KA#), as the "#'th" part for the private key, if any,

where the '#' characters are replaced by the number of parts. A reasonable minimum security recommendation is to let #≥2. The individual fields "SerialNo" are actually different values in

each part of a configuration control record. The symbol "¦" represents field concatenation according to some unambiguous syntax. The symbol ⊕ represents the "exclusive or operation". The function "Obfusc(...)" represents the "obfuscation" described below. The function "$PROT_K$(X¦Y¦Z¦P¦Q ...)" represents the cryptographic protection (integrity followed be encryption) of fields X, Y, Z, P, Q, ... (the configuration control record contents) using secret key K. As a security precaution, the first three such fields (X, Y, Z) repeat the cleartext record type tag, the optional indication of the public key, and the arbitrary record identifier. The fields starting with the fourth one (P, Q ...) are called the "payload" of the configuration control record. The payload may be empty. The details of the configuration control record contents in the above example (record type 'A') are explained below.

Here is a representative example for a configuration control record with an associated "key exchange key":

- SerialNo¦'C1#'¦PubK¦RecId¦Obfusc(KC1)¦$PROT_{KEK_{BK} \oplus KC1 \oplus KC2 \oplus ... \oplus KC\#}$('C#'¦PubK¦RecId¦KBK) as the first part for KBK,
- SerialNo¦'C2#'¦PubK¦RecId¦Obfusc(KC2), as the second part for KBK, if any,
- ...
- SerialNo¦'C##'¦PubK¦RecId¦Obfusc(KC#), as the "#'th" part for KBK, if any,

where KEK_{BK} is the independent "key exchange key" that protects the configuration control record contents. When there is a "key exchange key", a reasonable security recommendation is to let #≥1. The details of the configuration control record contents in the above example (record type 'C') are explained below.

With at least one record type, there is no actual contents in the configuration control record and the obfuscated secret key components are kept for the sole purpose of recovering a "Local Master Key" (LMK). Then, the storage of the LMK in multiple components uses nearly

identical conventions as with other record types:

- SerialNo¦'D1#'¦RecId¦Obfusc(LMK1)
- SerialNo¦'D2#'¦RecId¦Obfusc(LMK2)
- ...
- SerialNo¦'D##'¦RecId¦Obfusc(LMK#)

where the LMK is LMK1⊕LMK2⊕...⊕LMK#, and where the '#' characters are replaced by the number of key components. A reasonable minimum security recommendation is to let #≥2. The usage of the record type 'D' is explained below.

The obfuscation algorithm applied to secret key components is based on the secrecy of the algorithm, and/or the secrecy of a fixed key. Either way, the protection offered by obfuscation would not necessarily resist an elaborate reverse engineering attack on e.g. the SCD **101**. The "Frogbit" is a semi-proprietary scheme applicable to the obfuscation in the present invention. The Frogbit semi-proprietary algorithm is specified in the Canadian Patent Application Serial No. 2,177,622 published on November 30, 1997. The Frogbit variant selection is secretly done by a manufacturer of the present invention (following the guidelines disclosed in said application 2,177,622), and applied below as a pair of cryptographic primitives for encryption and decryption. Both the Frogbit variant programming and some fixed secret key are hidden into the key import processor **116** and into the microcontroller that drives the key component write interface **137**. For the present invention, the Frogbit scheme application may be as follows, for the writing of a secret key component:

1 a secret random binary string is chosen, e.g. 64 bits long,

2 a conventional CRC (Cyclic Redundancy Check) value (e.g. CRC-32 or preferably a custom CRC computation based on a polynomial of degree 64 or more) is computed on the following data elements: the media serial number, the record type tag (three characters), the optional indication of the public key, the arbitrary record identifier, and

the above secret random binary string,

3 a three-part ciphertext is computed as the Frogbit variant encryption of the following three data elements in sequence: the above random binary string, the above computed CRC value, and the secret key component, and

4 the obfuscated secret key component is made of the first part and third part of this three-part ciphertext (the second part is indeed omitted).

Reading a secret key component is the reverse process, that is:

1 the Frogbit variant decryption of the first part, giving the above random binary string,

2 the computation of the above CRC value,

3 the Frogbit variant encryption of the CRC value just computed (this step makes up for the above omission of the ciphertext second part),

4 the Frogbit variant decryption of the third part, giving the secret key component, but only if the reading operation is done in the identical context as the one in which the writing operation was done, that is

a) the same media serial number,

b) the same record type tag (three characters),

c) the same optional indication of the public key,

d) the same arbitrary record identifier, and

e) the same Frogbit variant implementation details.

It is advantageous that the obfuscated secret key component does not by itself reveal whether the de-"obfuscated" secret key component is valid or not. It is also advantageous that the media serial number is used in an "all-or-nothing" scheme, so that the production of a counterfeit media would require duplication of a serial number in an identical media package (the iButton form factor in the case of the preferred embodiment of the present invention).

Moreau, Thierry, *Cryptographic Data Integrity Apparatus and Method Based on*

Pseudo-Random Bit Generators, Canadian patent application number 2,177,622, filed on May 29, 1996

List of key management operations and record types.

Operation PPKP generation

Let the KGD create a new PPKP (creates records 'A', and 'b', and technology obsolescence recovery media for PPKP), without user access restrictions.

Record type 'A'

Read by: SPKCA

Associated independent secret key: none

Payload: PubKObjId¦PPKP¦KEK_{BK}¦Flag_B¦Flag_C

Note: The flags indicate whether the list of KEK_{BK} in the SPKCA will be erased upon some circumstances as follows:

Flag_B: receipt of the next 'B' configuration control record,

Flag_C: receipt of any 'C' configuration control record following the receipt of this 'A' configuration control record.

Record type: 'b'

Read by: KGD

Associated independent secret key: none

Payload: KEK_{BK}

Usage: For the current KEK_{BK}

Operation KEK_{BK} update preparation

Let the KGD prepare updating the KEK_{BK}, starting from 'b' (either the original 'b', that is the one created along with the 'A' configuration control record, or the current 'b', that is the most recent successful KEK_{BK} update operation), producing brand new 'B' and 'b'.

Record type: 'B'

Read by: SPKCA

Associated independent secret key: $OldKEK_{BK}$

Payload: $NewKEK_{BK}$¦Flag_0¦Flag_B¦Flag_C

Usage: for the renewal of KEK_{BK}

Note: The flags indicate whether the complete list of KEK_{BK} in the SPKCA will be erased upon some circumstances as follows:

Flag_0: receipt of this 'B' configuration control record,

Flag_B: receipt of the next 'B' configuration control record,

Flag_C: receipt of any 'C' configuration control record following the receipt of this 'B' configuration control record.

The list of KEK_{BK} in the SPKCA will be erased upon receipt of a 'B' record if Flag_0 is set or if Flag_B has been set in the previous 'A' or 'B' record.

Note: The notation $OldKEK_{BK}$ and $NewKEK_{BK}$ refer to the fact that 'B' records update the key KEK_{BK} using its current value.

Operation KBK injection preparation

Let the KGD prepare installing or updating KBK, starting from 'b', producing brand new 'C' and 'c'.

Record type: 'C'

Read by: SPKCA

Associated independent secret key: KEK_{BK}

Payload: KBK

Usage: to inject a new value for KBK

Record type: 'c'

Read by: KGD

Associated independent secret key: none

Payload: KBK

Usage: for the use of a SPKCA's KBK

Operation LMK initialization

Initialize the KGD for the management of KEK_{FA} list, creating some LMK components ('D').

Record type: 'D'

Read by: KGD

Associated independent secret key: none

Payload: n/a, this record type is devoid of any cryptogram of a configuration control record (the format of the components for the 'D' record type has been explained above)

Usage: used locally by the KGD to protect the database of KEK_{FA}

Operation LMK re-installation

Re-install the KGD for the management of KEK_{FA} list, from some existing LMK components ('D').

Operation LMK session start

Start a KGD user session for the management of KEK_{FA} list, e.g. from some existing LMK components ('D') and user authorization validation.

Operation LMK session finish

Finish a KGD user session for the management of KEK_{FA} list.

Operation KEK_{FA} list update preparation

In the course of a KGD user session for the management of KEK_{FA} list, let the KGD prepare updating an SPKCA configuration for KEK_{FA} list, starting from 'c', producing maybe 'E' and one or more 'F' and 'G'.

Record type: 'E'

Read by: SPKCA

Associated independent secret key: KBK

Payload: empty string

Usage: to delete every KEK_{FA} from the SPKCA

Record type: 'F'

Read by: SPKCA

Associated independent secret key: KBK

Payload: FAid¦KEK_{FA}

Usage: to install a new KEK_{FA} or to renew one

Record type: 'G'

Read by: SPKCA

Associated independent secret key: KBK

Payload: FAid

Usage: to delete a KEK_{FA} from the SPKCA

When a device (SPKCA or KGD) receives a configuration control record in multiple parts, they can simply be presented to the device consecutively in any order.

Operation SPKCA configuration update

The receipt of configuration messages by the SPKCA ('A', 'B', 'C', 'E', 'F', and 'G'). The receipt of configuration messages is allowed only when the SPKCA 4-position switch 111 is in "setup" position.

Operation "hello message" generation

The spontaneous generation (by the SPKCA) of occasional "hello messages" that reflect the current configuration. The spontaneous generation (by the SPKCA) of "hello messages" occurs when the SPKCA 4-position switch 111 is in the "standby" position or in the "operating" position. When the SPKCA 4-position switch 111 is in "standby" position, hello messages are generated at an higher pace than when the SPKCA 4-position switch 111 is in the "operating" position.

Operation SPKCA zeroization

Complete erasure of the SPKCA configuration. This operation is forced whenever the SPKCA 4-position switch 111 is brought to the "program load" position, upon detection of external battery power outage, and upon detection of tampering with the physical integrity of the SCD 101.

Operation PRNG state initialization

To inject a pseudo-random number generator (PRNG) state in the SPKCA.

The "hello messages" are provided in the present invention to allow monitoring, troubleshooting and operational diagnostic of an SPKCA which is otherwise very discreet about

its internal state. Logically, each PPKP present in the SPKCA configuration should trigger one "hello message" and each KEK_{FA} should also trigger one "hello message". With this arrangement, there is an upper bound in the size of an "hello message" irrespective of the number of PPKP and KEK_{FA} present in the SPKCA configuration. The "hello message" for a PPKP should include:

- the reference "PubK" to the PPKP public key counterpart,
- the object identifier, "PubKObjId" that identifies the PPKP on a more global scope,
- the actual value of the PPKP public key counterpart,
- the check digits for the current KEK_{BK},
- the check digits for the current KBK.

The "hello messages" for a KEK_{FA} should include:

- the reference "PubK" to the PPKP public key counterpart,
- the reference FAid to the functional application area,
- the check digits for the current KEKFA.

By monitoring what goes out of the SPKCA through the output network connection point 110, it is possible to gather a complete image of the SPKCA configuration over the period required for the SPKCA to cycle through every element in its current configuration.

The synchronization of configuration between the KGD and the one or more SPKCAs that are managed through the KGD is partly a matter of operational discipline: physical access control to the SPKCA, meticulously following the instructions about presentation of SPKCA configuration control records, destruction or return of configuration control records. The main threats to be managed in this context might be a) the delays in the removal of a KEK_{FA} once an FA system has been found to be compromised, or worse, the re-enabling of a KEK_{FA} some time

after its removal, and b) the undetected theft of a complete set of configuration control records, followed by the theft of an SPKCA device. Both threats may be managed by renewing the keys KEK_{BK} and/or KBK (respectively 'B' and 'C'). These renewal operation might not invalidate the current list of KEK_{FA} in the SPKCA (based upon flag indicators in the 'A' or 'B' configuration control records). In the case of the key KEK_{BK}, the control of the renewal operation normally requires the collaboration of more than one key custody. The first threat (delayed removal of KEK_{FA}) can also be managed by monitoring the "hello messages" produced by the SPKCA device.

With the key management principles explained herein, the present invention allows disaster recovery strategies that do not expose the private counterpart of the PPKP to undue threats. For instance, a spare SPKCA may be set up in the disaster recovery facilities with the live PPKP, and a dedicated value for the key KEK_{BK}. The spare SPKCA remains under the same operational control as the disaster recovery facilities but the safe storage (e.g. in the corporate safe boxes) of the dedicated KEK_{BK} components prevents its misuse. The database of KEK_{FA}'s and the corresponding LMK secret key should be backed up respectively like other normal operational data and like cryptographic key material used for "usual" information security applications. Typically, the applications that depend on the SPKCA operations would not be among the very first priorities upon occurrence of a disaster: in the case of a certification authority's private signature key, the routine verification of security certificates does not depend on the SPKCA, while in the case of secret key establishment, the SPKCA typically used only at registration time for new customers in some secure on-line services, so existing customers can be served without the SPKCA. Thus it is reasonable to plan disaster recovery as follows:

1 grab any KGD and the backup copies of the database of KEK_{FA}'s and the LMK,

2 re-install any required KEK_{FA} in the FA systems that might have been lost in the disaster,

3 have the key custodians the for the dedicated KEK_{BK} come to the KGD in order to trigger the generation of a new KBK, that is configuration control record types 'C' and

'c',

4 generate the configuration control record types 'E' and 'F' that allows injection of the KEK_{FA}'s into the SPKCA,

5 inject into the SPKCA the new KBK, and the KEK_{FA}'s (record types 'C', 'E', and 'F').

5 While the present invention has been disclosed and described with reference to a preferred embodiment, it will be apparent as noted above that various changes, alterations, and substitutions can be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1 A secure cryptographic device implementing a public key cryptography primitive with a one-way input connection and a one-way output connection.

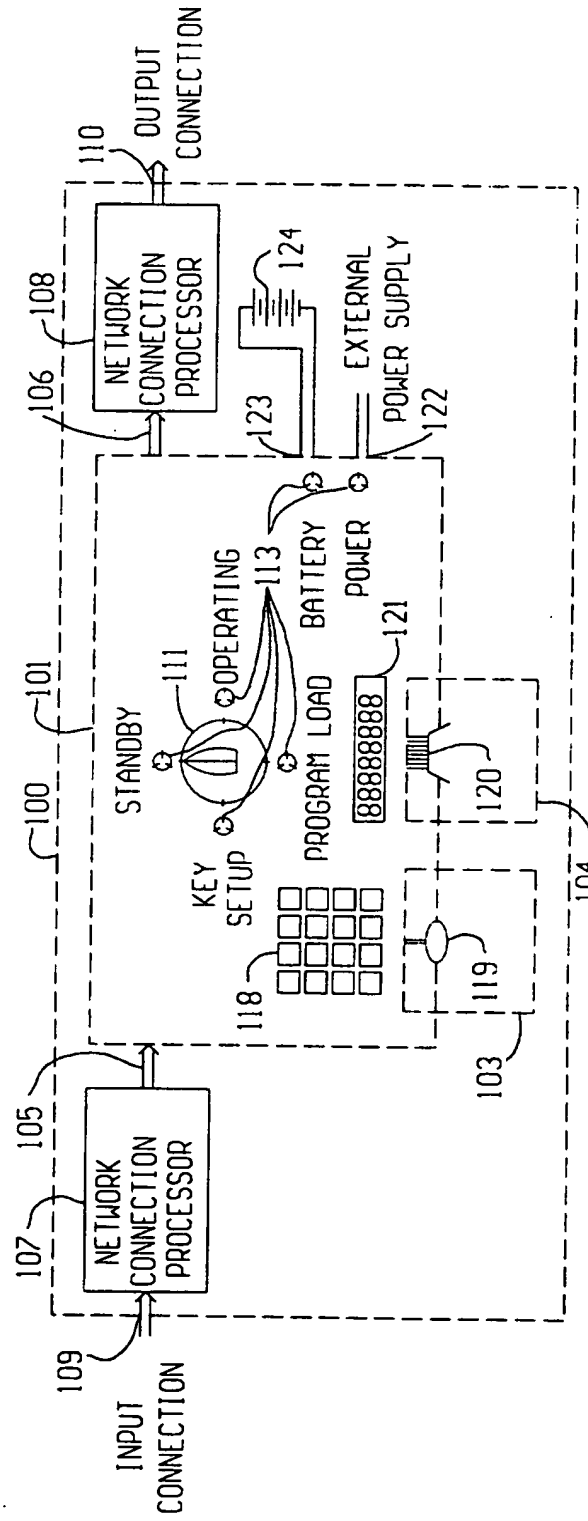

OUTPUT CONNECTION
110
NETWORK CONNECTION PROCESSOR
108
106
124
EXTERNAL POWER SUPPLY
122
123
POWER
BATTERY
113
OPERATING
111
STANDBY
101
100
PROGRAM LOAD
121
88888888
KEY SETUP
118
120
119
104
103
105
NETWORK CONNECTION PROCESSOR
107
109
INPUT CONNECTION

Figure 1

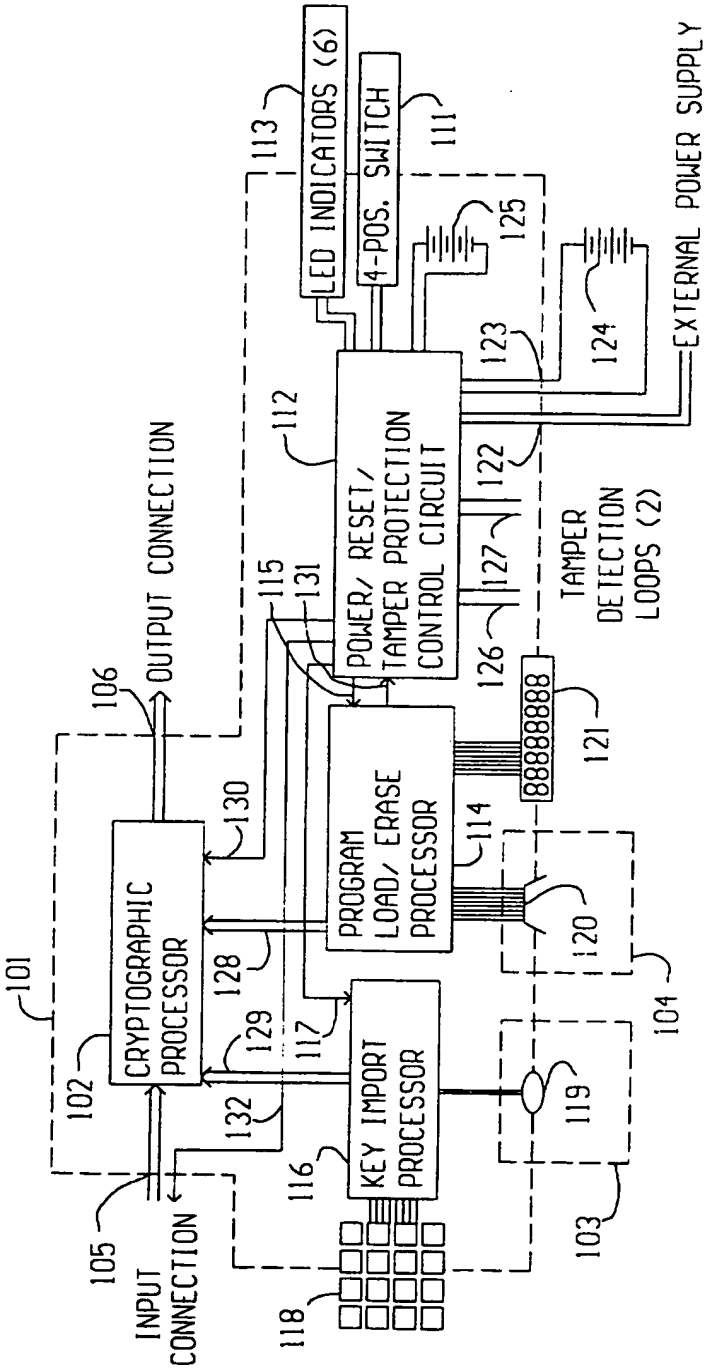

101
102
CRYPTOGRAPHIC PROCESSOR
106
OUTPUT CONNECTION
105
INPUT CONNECTION
116
KEY IMPORT PROCESSOR
118
PROGRAM LOAD/ ERASE PROCESSOR
114
POWER/ RESET/ TAMPER PROTECTION CONTROL CIRCUIT
112
LED INDICATORS (6)
113
4-POS. SWITCH
111
EXTERNAL POWER SUPPLY
TAMPER DETECTION LOOPS (2)
88888888
121
103
104
119
120
123
124
125
122
126
127
128
129
130
131
115
117
132

Figure 2

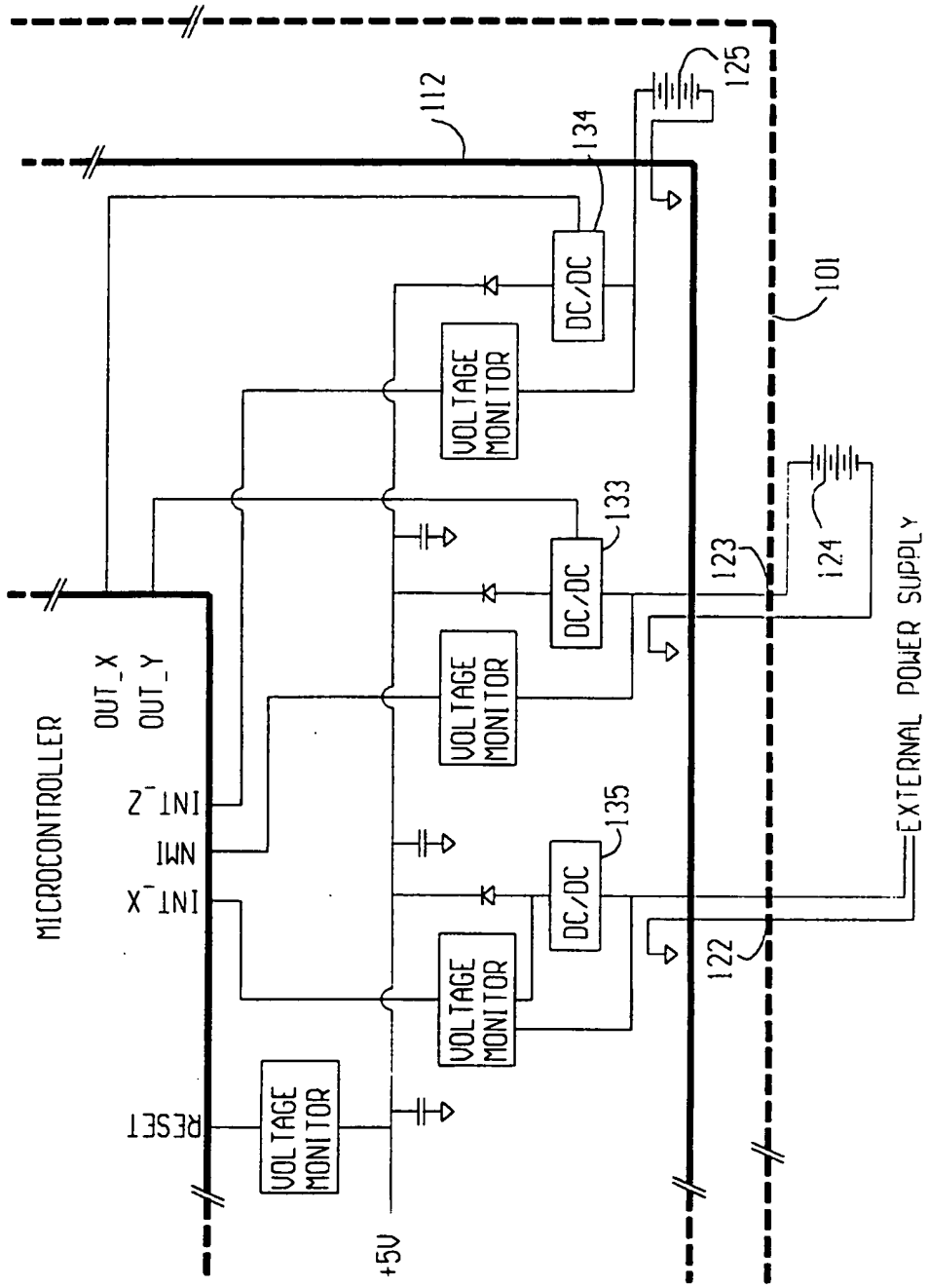

MICROCONTROLLER
OUT_X
OUT_Y
INT_Z
NMI
INT_X
RESET
+5V
VOLTAGE MONITOR
VOLTAGE MONITOR
VOLTAGE MONITOR
VOLTAGE MONITOR
DC/DC
DC/DC
DC/DC
112
134
133
135
125
101
123
124
122
EXTERNAL POWER SUPPLY

Figure 3

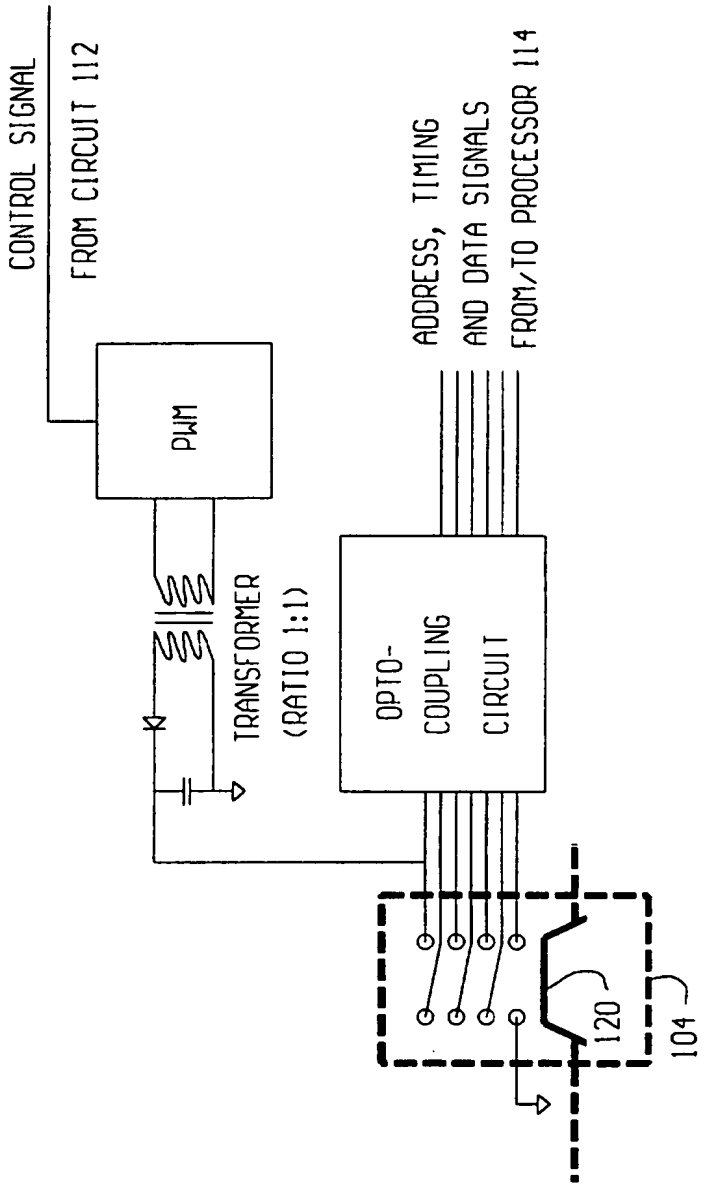

CONTROL SIGNAL
FROM CIRCUIT 112
PWM
TRANSFORMER
(RATIO 1:1)
OPTO-
COUPLING
CIRCUIT
ADDRESS, TIMING
AND DATA SIGNALS
FROM/TO PROCESSOR 114
120
104

Figure 4

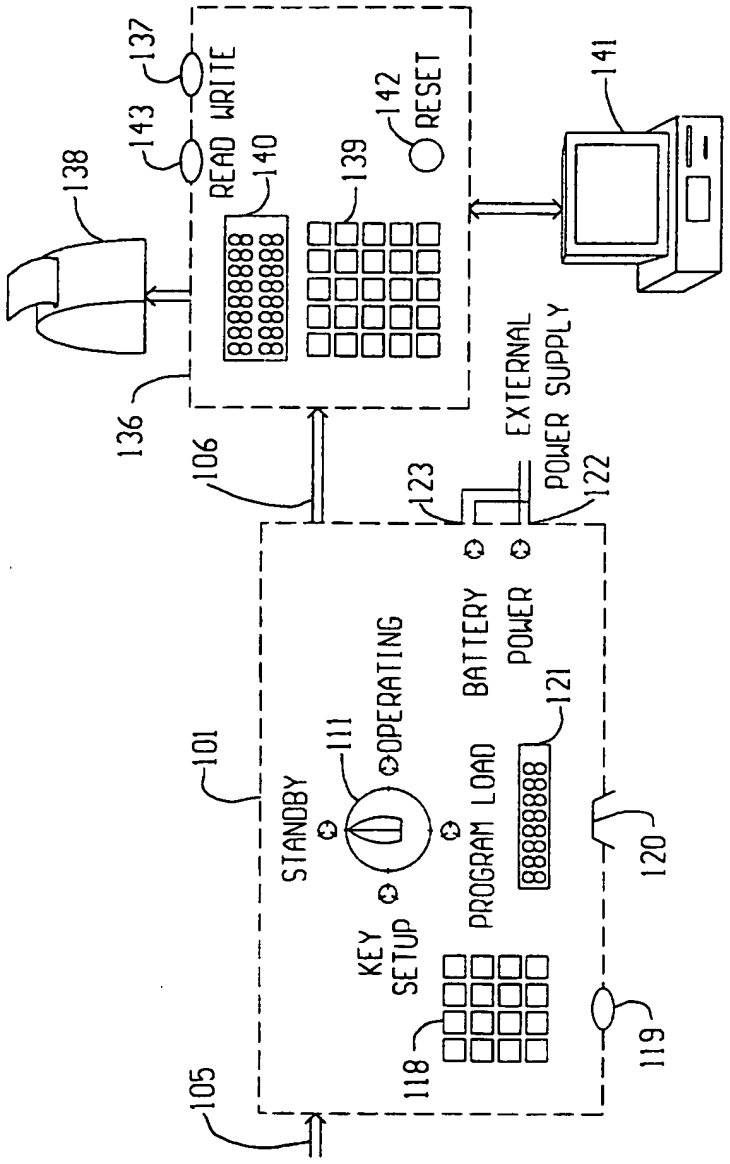


Figure 5